Object Oriented Programming with Java 8

Dr.Akshita Chanchlani

Trainer Introduction

- **Name**: Dr.Akshita S.Chanchlani
- **Designation** : Associate Head- Technical
- **Education** :
  - PhD : Computer Science and Engineering
  - Masters of Engineering (ME) in Information Technology
  - B.Tech in Computer Engineering, From VJTI Mumbai
- **Training Experience**
  - PreCAT : C, C++, Operating System, Networks, AI
  - PG Course : Core Java, Python, Data Structure using Java
  - Modular Batches : Core Java, Python, Machine Learning
  - Internship with Live Project : Machine Learning
- **Professional Experience**
  - **13+ years**
- **Email** : akshita.chanchlani@sunbeaminfo.com

Day01 Agenda

- Documentation
- Java history
- Version history
- Java platforms , SDK, JDK, JRE, JVM
- Components of Java platform
- C++ versus Java terminology
- Core Java terminologies
- Structure of Java class
- Java application flow of execution
- Comments and Entry point method

Java Text Books

- Java, The complete reference, Herbert Schildt
- Core and Advanced Java Black Book
- Head First Java: A Brain-Friendly Guide , by Kathy Sierra

Documentation

- **JDK download:**
  - https://adoptopenjdk.net/
- **Spring Tool Suite 3 download:**
  - https://github.com/spring-projects/toolsuite-distribution/wiki/Spring-Tool-Suite-3
- **Oracle Tutorial:**
  - https://docs.oracle.com/javase/tutorial/
- **Java 8 API and Java 11 API Documentation Documentation:**
  - https://docs.oracle.com/javase/8/docs/api/
  - https://docs.oracle.com/en/java/javase/11/docs/api/
- **MySQL Connector:**
  - https://downloads.mysql.com/archives/c-j/
- **Core Java Tutorials:**
  1. http://tutorials.jenkov.com/java/index.html
  2. https://www.baeldung.com/java-tutorial
  3. https://www.journaldev.com/7153/core-java-tutorial

Java Installation

1. JDK 11 download
https://www.oracle.com/java/technologies/javase-jdk11-downloads.html
Choose your platform , download the installer and install the JDK

2. Open command prompt or terminal
Type
java -version
It should show you : java version "11.0.7" 2020-04-14 LTS
Note : Basic version should be 11, update version may differ.

3.Java IDE download
https://github.com/spring-projects/toolsuite-distribution/wiki/Spring-Tool-Suite-3
Choose 3.9.16 version and download. Extract it. No installation is required.

4. Java API Dcoumentation link
https://docs.oracle.com/en/java/javase/11/docs/api/index.html

5. Offline version
https://www.oracle.com/java/technologies/javase-jdk11-doc-downloads.html

Java History

- **James Gosling, Mike Sheridan** and **Patrick Naughton** initiated the Java language project in June 1991.
- **Java** was originally **developed by James Gosling** at **Sun Microsystems** and **released in 1995.**
- The language was initially called **Oak** after an oak tree that stood outside Gosling's office.
- Later the project went by the name ***Green*** and was finally renamed ***Java***, from Java coffee, a type of coffee from Indonesia.
- Gosling and his team did a brainstorm session and after the session, they came up with several names such as **JAVA, DNA, SILK, RUBY, etc.**
- Sun Microsystems released the first public implementation as Java 1.0 in 1996.

Java History

- The Java programming language is a general-purpose, concurrent, class-based, object-oriented language.
- The Java programming language is a high-level language.
- The Java programming language is related to C and C++ but is organized rather differently, with a number of aspects of C and C++ omitted and a few ideas from other languages included.
- **It is intended to be a production language, not a research language.**
- The Java programming language is statically typed.
- It promised **Write Once, Run Anywhere** (**WORA**) functionality.

Version History

Version	Date
JDK Beta	1995
JDK1.0	January 23, 1996[39]
JDK 1.1	February 19, 1997
J2SE 1.2	December 8, 1998
J2SE 1.3	May 8, 2000
J2SE 1.4	February 6, 2002
J2SE 5.0	September 30, 2004
Java SE 6	December 11, 2006
Java SE 7	July 28, 2011
Java SE 8	March 18, 2014
Java SE 9	September 21, 2017
Java SE 10	March 20, 2018
Java SE 11	September 25, 2018[40]
Java SE 12	March 19, 2019
Java SE 13	September 17, 2019
Java SE 14	March 17, 2020
Java SE 15	September 15, 2020[41]
Java SE 16	March 16, 2021

- The first version was released on January 23, 1996.

- The acquisition of Sun Microsystems by Oracle Corporation was completed on January 27, 2010

- As of September 2020, Java 8 and 11 are supported as Long Term Support (LTS) versions

- In September 2017, Mark Reinhold, chief Architect of the Java Platform, proposed to change the release train to "one feature release every six months".

- OpenJDK (Open Java Development Kit) is a free and open source implementation of the (Java SE). It is the result of an effort Sun Microsystems began in 2006.

- Java Language and Virtual Machine Specifications: https://docs.oracle.com/javase/specs/

Java Platforms

1. **Java SE**
   - Java Platform Standard Edition.
   - It is also called as Core Java.
   - For general purpose use on Desktop PC's, servers and similar devices.
2. **Java EE**
   - Java Platform Enterprise Edition.
   - It is also called as advanced Java / enterprise java / web java.
   - Java SE plus various API's which are useful client-server applications.
3. **Java ME**
   - Java Platform Micro Edition.
   - Specifies several different sets of libraries for devices with limited storage, display, and power capacities.
   - It is often used to develop applications for mobile devices, PDAs, TV set-top boxes and printers.
4. **Java Card**
   - A technology that allows small Java-based applications (applets) to be run securely on smart cards and similar small-memory devices.

WHY Java ?

1. Platform or architecture independent (Write once run anywhere!)
2. Simple & robust
3. Secure
4. Automatic memory management.
5. Inherent Multi threaded support
6. Object Oriented support -- Encapsulation, Inheritance & polymorphism , abstraction.
7. Excellent I/O support
8. Inherent networking support for TCP/IP , UDP/IP programming & for URLs
9. Adds functional programming support.

SDK, JDK, JRE, JVM

- **SDK** = Development Tools + Documentation + Libraries + Runtime Environment.

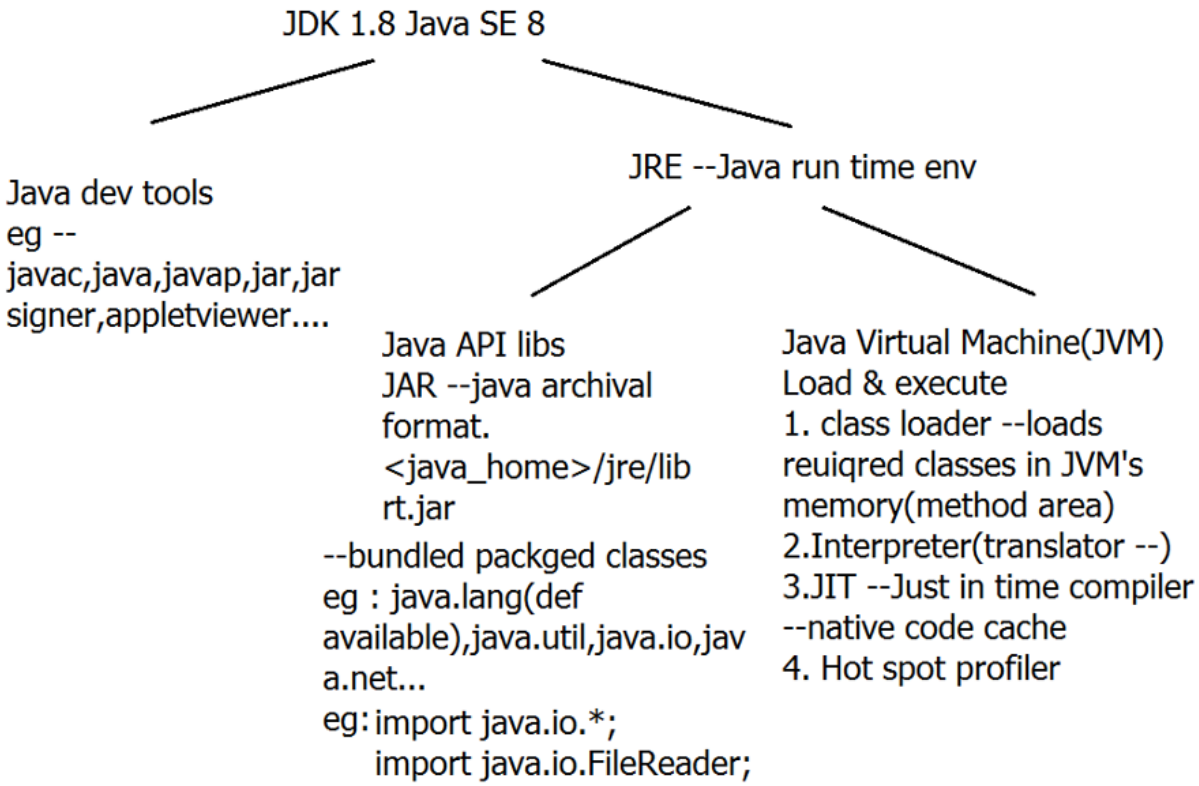
- **JDK** = Java Development Tools + Java Docs + **rt.jar** + JVM.
  - JDK : Java Development Kit.
  - It is a software, that need to be install on developers machine.
  - We can download it from oracle official website.

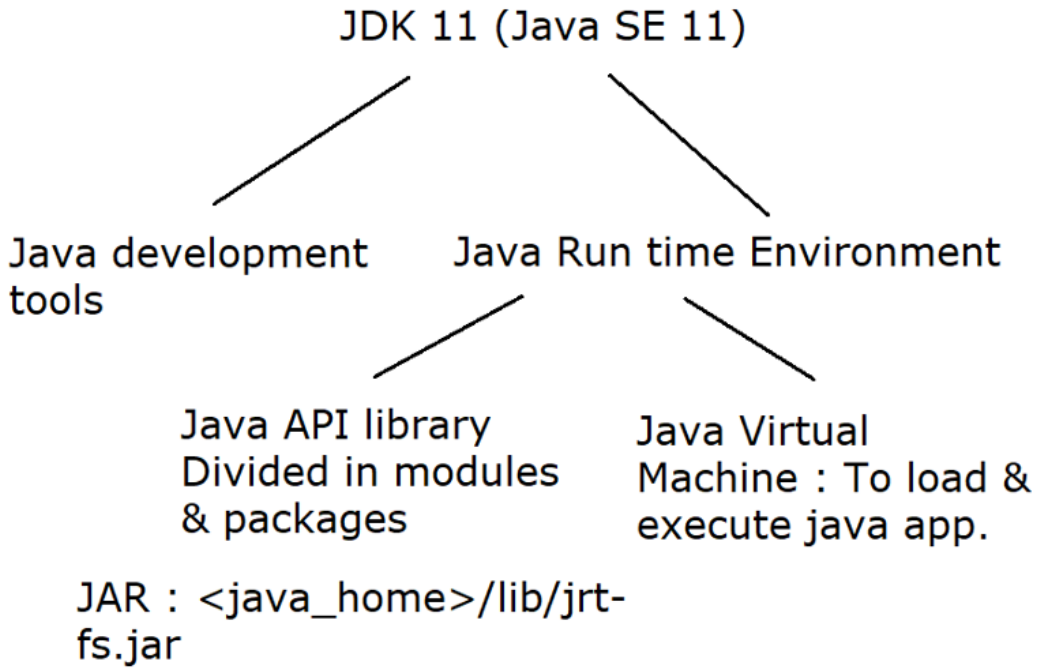
- **JDK** = Java Development Tools + Java Docs + **JRE[rt.jar + JVM**].
  - JRE : Java Runtime Environment.
  - rt.jar and JVM are integrated part of JRE.
  - JRE is a software which comes with JDK. We can also download it separately.
  - To deploy application, we should install it on client's machine.

- rt.jar file contains core Java API in **compiled form**.

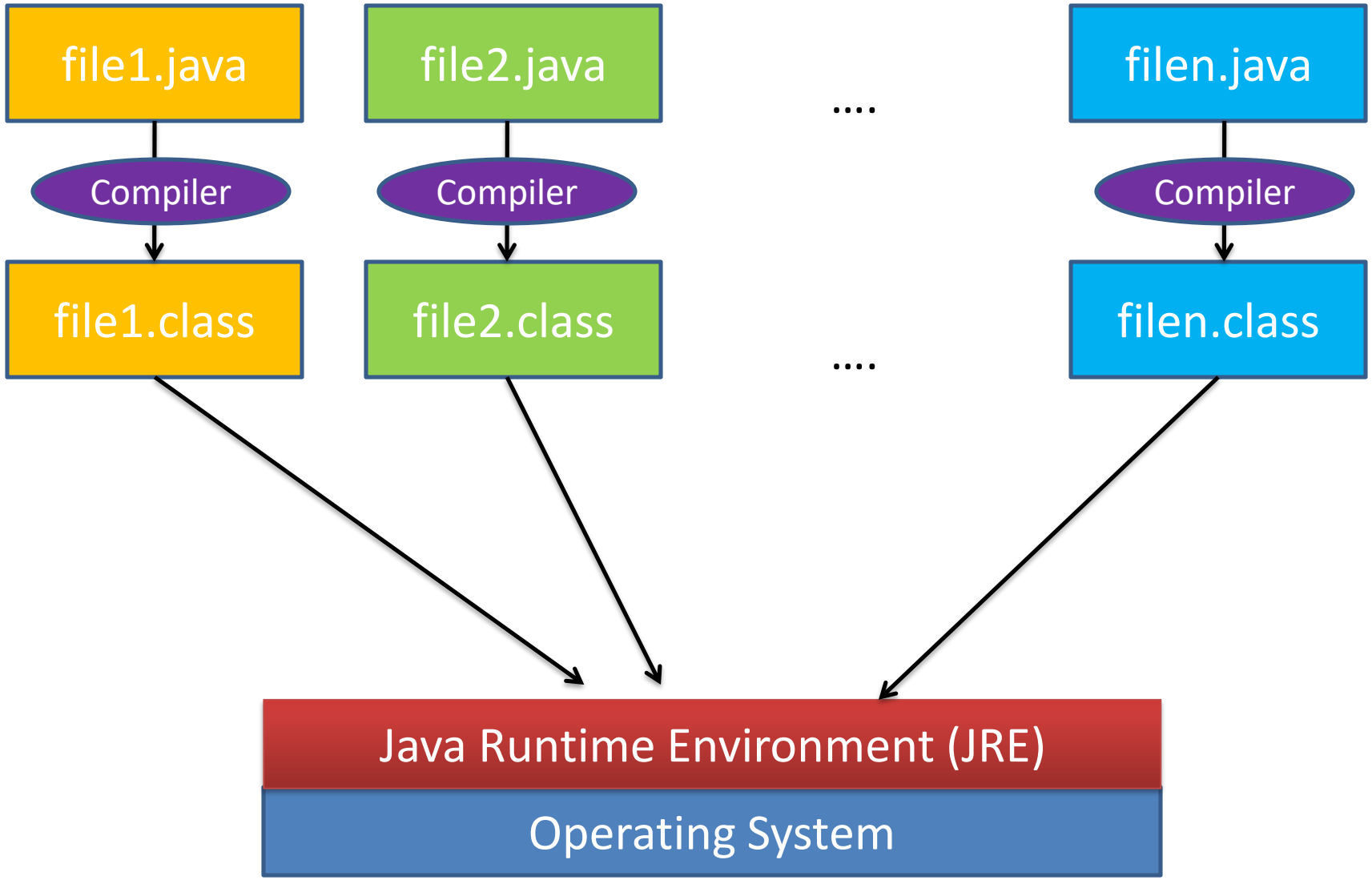
- **JVM** : An engine, which manages execution of Java application. (also called as Execution Engine)

JDK


JDK 1.8 Java SE 8
Java dev tools
eg --
javac,java,javap,jar,jar signer,appletviewer....
JRE --Java run time env
Java API libs
JAR --java archival format.
<java_home>/jre/lib rt.jar
--bundled packged classes
eg : java.lang(def available),java.util,java.io,java.net...
eg: import java.io.*;
import java.io.FileReader;
Java Virtual Machine(JVM)
Load & execute
1. class loader --loads reuiqred classes in JVM's memory(method area)
2.Interpreter(translator --)
3.JIT --Just in time compiler
--native code cache
4. Hot spot profiler


JDK 11 (Java SE 11)
Java development tools
Java Run time Environment
Java API library
Divided in modules & packages
Java Virtual Machine : To load & execute java app.
JAR : <java_home>/lib/jrt-fs.jar

Java compiler usage



C++ versus Java terminology

C++ Terminology	Java Terminology
Class	Class
Data Member	Field
Member Function	Method
Object	Instance
Pointer	Reference
Access Specifier	Access Modifier
Base Class	Super Class
Derived Class	Sub Class
Derived From	Extended From
Runtime Polymorphism	Dynamic Method Dispatch

C++ VS Java

Development wise differences

- Java is platform independent language but c++ is dependent upon operating system. At compilation time Java Source code(.java) converts into byte code(.class) .The interpreter translates this byte code at run time into native code and gives output.
- Java uses both a compiler and interpreter, while C++ only uses a compiler

Syntactical differences

- 1. There is no final semi-colon at the end of the class definition.
- 2. Functions are called as methods.
- 3. main method is a member of class & has a fixed form
  - public static void main(String[] args) -- argument is an array of String. This array contains the command-line arguments.
- 4. main method must be inside some class (there can be more than one main function -- there can even be one in every class)
- 5. Like the C++ << operator,
  - To write to standard output, you can use either of the following:
    - System.out.println(...)
    - System.out.print(...) The + operator can be useful when printing.

C++ VS Java cont..

Features wise differences

- 1. C++ supports pointers whereas Java does not support pointer arithmetic.
- 2. C++ supports operator overloading , multiple inheritance but java does not.
- 3. C++ is nearer to hardware than Java.
- 4. Everything (except fundamental or primitive types) is an object in Java (Single root hierarchy as everything gets derived from java.lang.Object).
- 5. Java is similar to C++ but it doesn't have the complicated aspects of C++, such as pointers, templates, unions, operator overloading, structures, etc. Java also does not support conditional compilation (#ifdef/#ifndef type).
- 6. Thread support is built into Java but not in C++. C++11, the most recent iteration of the C++ programming language, does have Thread support though.
- 7. Internet support is built into Java, but not in C++. On the other hand, C++ has support for socket programming which can be used.
- 8.Java does not support header files and library files. Java uses import to include different classes and methods.
- 9.Java does not support default arguments.
- 10. There is no scope resolution operator :: in Java. It has . (dot operator) using which we can qualify classes with the namespace they came from.
- 11. There is no goto statement in Java.
- 12. Because of the lack of destructors in Java, exception and auto garbage collector handling is different than C++.
- 13. Java has method overloading, but no operator overloading unlike C++.
- 14. The String class does use the + and += operators to concatenate strings and String expressions use automatic type conversion,
- 15. Java is pass-by-value.
- 16. Java does not support unsigned integers.

Language Basics

- Keywords
- Variables
- Conditional Statements
- Loops
- Data Types
- Operators
- Coding Conventions

Java Keywords

abstract	boolean	break	byte	case	catch
char	class	const	continue	default	do
double	else	extends	final	finally	float
for	goto	if	implements	import	instanceof
int	interface	long	native	new	package
private	protected	public	return	short	static
strictfp	super	switch	synchronized	this	throw
throws	transient	try	void	volatile	while
true	false	null			

Simple Hello Application

```
//File Name : Program.java
class Program{
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

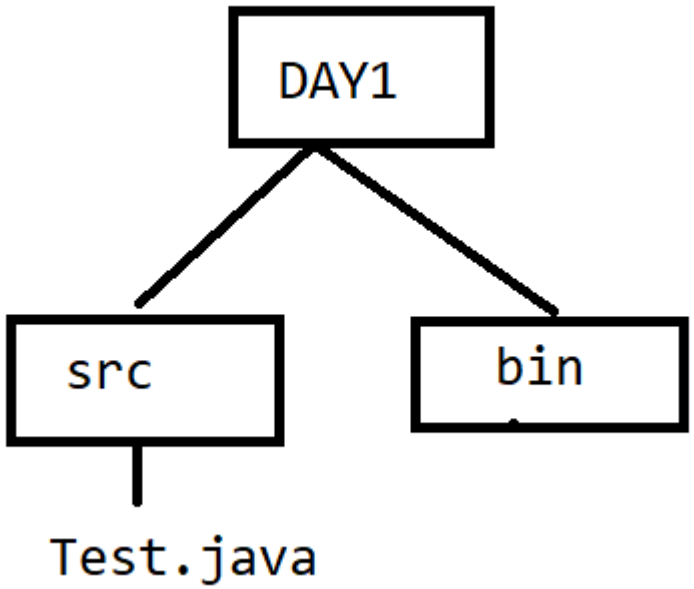
```
Compilation=> javac Program.java  //Output : Program.class
Execution=>   java Program
```

```
System    :   Final class declared in java.lang package
out       :   public static final field of System class. Type of out is PrintStream
println   :   Non static method of java.io.PrintStream class
```

To view class File :

javap -c Program.class

How to compile the code from src and store .class inside bin




```
D:\DAY1> cd src
D:\DAY1\src>javac –d ..\bin Test.java
D:\DAY1\src>cd ..\bin
D:\DAY1\bin>java Test
```

java.lang.System class

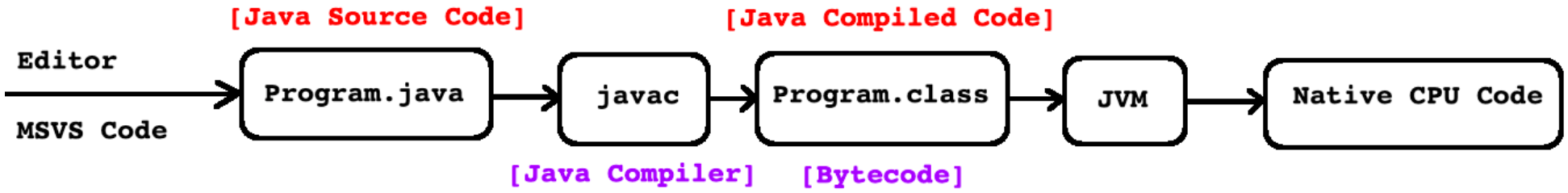
```
package java.lang;
import java.io.*;
public final class System{
    public static final InputStream in;
    public static final OutputStream out;
    public static final OutputStream err;

    public static Console console();
    public static void exit(int status);
    public static void gc();
}
```

Stream

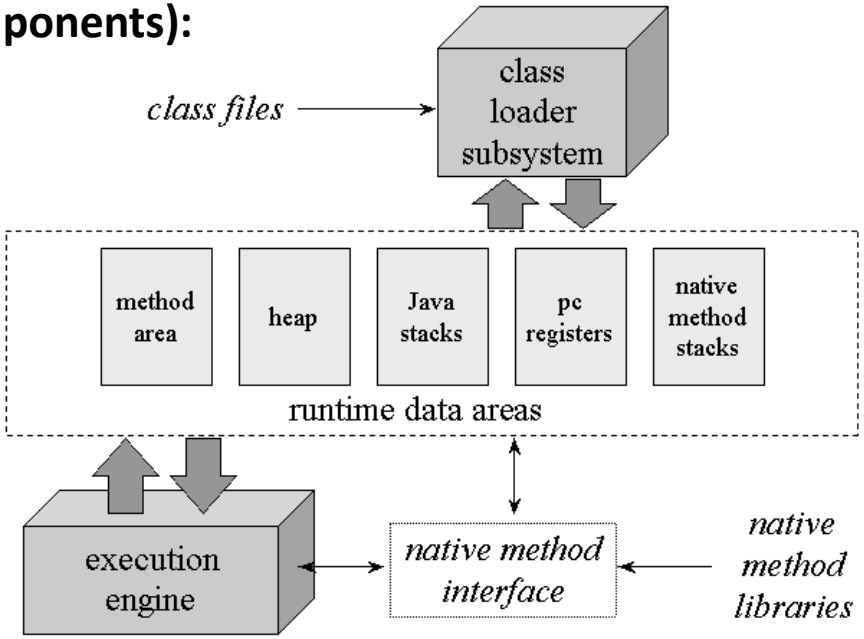
- Stream is an abstraction(object) which either produce(write)/consume(read) information from source to destination.
- Standard stream objects of Java which is associated with console:
    1. **System.in**
        - Ø It represents keyboard.
    2. **System.out**
        - Ø It represents Monitor.
    3. **System.err**
        - Ø Error stream which represents Monitor.

Java application flow of execution



Components of JVM (Major 3 Components):

1. **Class Loader Sub System**
   - Bootstrap class loader
   - Extension classLoader
   - Application classLoader
   - User Defined class Loader
2. **Runtime Data Areas**
3. **Execution Engine**
   - Interpreter
   - Compiler
   - Garbage Collector



[JVM Architecture]

Method Area

- Metaspace: JDK 1.8 onwards Loaded Byte codes (Loaded class info) 1 Single copy static data members, constructors, methods

Heap

- Java object heap state of the object (non static data members) 1 single copy

Java Stack

- Stack is created one per thread , method local info(like args,local vars, ret vals) Individual stack : stack frames

Comments

- Comments should be used to give overviews of code and provide additional information that is not readily available in the code itself. Comments should contain only information that is relevant to reading and understanding the program.
- Types of Comments:
  - **Implementation Comment**
    - Implementation comments are those found in C++, which are delimited by /*...*/, and //.
      1. Single-Line Comment
      2. Block Comment(also called as multiline comment)
  - **Documentation Comment**
    - Documentation comments (known as "doc comments") are Java-only, and are delimited by **/**...*/**.
    - Doc comments can be extracted to HTML files using the javadoc tool.

Entry point method

- Syntax:
  1. **public static void main(String[] args)**
  2. **public static void main(String... args)**

- Java compiler do not check/invoke main method. JVM invoke main method.

- When we start execution of Java application then JVM starts execution of two threads:
  1. Main thread : responsible for invoking main method.
  2. Garbage Collector : responsible for deallocating memory of unused object.

- We can overload main method in Java.

- We can define main method per class. But only one main method can be considered as entry point method.

Thank you.

akshita.Chanchlani@sunbeaminfo.com